THE FAIL-STOP CONTROLLER AE11

E. Böhl, Th. Lindenkreuz, R. Stephan

Robert Bosch GmbH — Automotive Equipment Division 8
72703 Reutlingen, Germany

Abstract

Using on-chip fault detection measures the Fail-Stop Controller AE11 was developed for safety critical applications aiming at high volume production of automotive and railway electronics. The trade-off between high defect coverage, short reaction time to faults and low chip area overhead results in a combination of Concurrent Checking, Built-In Self-Test and Built-In Current-Monitoring (I_{DDQ}-Test).

1 Introduction

The Fail-Stop Controller (FSC) was developed by Bosch, Alcatel/SEL, Siemens, Porsche, SGS-Thomson and the Universities of Hannover, Erlangen-Nürnberg and Grenoble. The goal was to replace the classic safety structure with duplicated data processing paths by a single controller with a considerably smaller hardware overhead and a safety level comparable with the classic solution. The solutions presented in [1], [2] and [3] have evolved from aims similar to ours - to detect hardware defects on-line. The area overhead of these solutions is estimated at about 100% and more for an 8-bit data bus. This is too much for high volume production since a duplicated controller structure (using two standard controllers) is less expensive. Therefore, our aim was to develop a special controller with considerably less overhead than the classic solution.

The FSC is able to flag the occurrence of hardware faults under all conditions. In such a case, the supervising system can stop operation, degrade gracefully or signal a fault condition (fail-safe system). Two different kinds of self-tests are in use. In Concurrent Checking (CC) methods which use coding techniques, the fault detection takes place at the moment of manifestation. In preventive detection the unit has a Built-In Self-Test (BIST) which tests the hardware structure during a special test mode phase. In order to cope with the high defect coverage requirements of the BIST, an on-line I_{DDQ}-Test is incorporated as a complementary test technique to the stuck-at self-test. It has been proven by fault simulation that the on-chip monitoring of the quiescent current in static CMOS gates (I_{DDQ}-Test) is an essential part of the BIST and constitutes an advanced test technique also for on-line test purposes.

In our applications special attention must be focused on the performance loss caused by delay for CC and by BIST/I_{DDQ} circuitry. Furthermore, self-test methods and, in particular, on-chip monitoring of the quiescent current in static CMOS gates (I_{DDQ}-Test), have a tremendous impact on the design flow and represent a challenging problem for the chip design. The purpose of this paper is to outline this project as a comprehensive approach to a customized controller for safety-critical applications.

This work was supported by the JESSI project AE11.

2 General Features of the Controller

The FSC was developed for the CMOSM5 technology of SGS-Thomson (0.7 µm, two metal layers) and has the following general features:

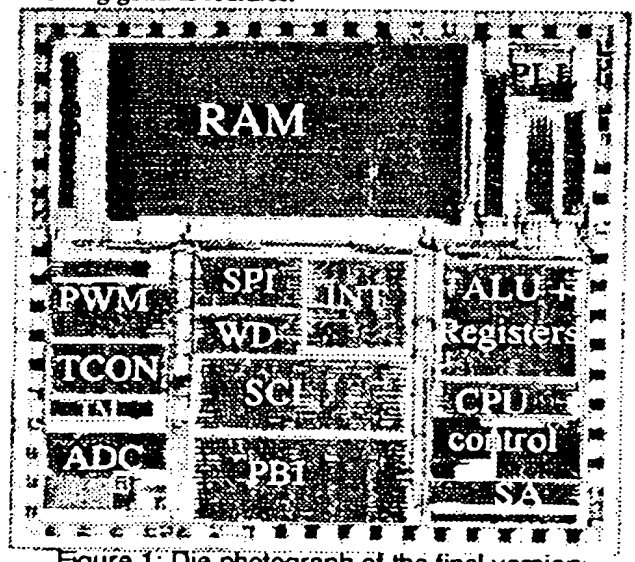

Figure 1: Die photograph of the final version

- 8 Bit CPU compatible with ECO51 (Siemens) and 8051 (Intel)
- 4 kByte RAM
- 8 Bit ADC with 8 inputs
- Serial Peripheral Interface (SPI)
- Serial Communication Interface (SCI)
- Serial Interface supporting the Profibus Protocol (PBI)
- Programmable Watchdog (WD)
- Pulse-Width-Modulation outputs (PWM)
- Support for program flow Signature Analysis (SA)
- Digital I/O ports
- Clock generator with PLL (internal typically 16 Mhz)
- External crystal clock 4 Mhz
- External parallel address/data bus (EBI)

0-7803-4209-7/97 $10.00 © 1997 IEEE

- Interrupt unit with 2 external and 6 internal sources
- Power supply monitor with detection of voltage overflow and underflow
- Temperature monitor (TM)
- Test controller (TCON) including JTAG and TAP
- Boundary Scan Path (BS)

3 Safety Architecture

The safety assessment is directly related to the fault coverage achieved by the on-chip fault detection measures. For railway and automotive applications a fault coverage of about 99% is necessary to fulfill the safety requirements as shown in Section 6.2.

Defects becoming effective during safety-critical operations must be detected within an application-specific time period to give the system a chance to react in time before a catastrophic malfunction takes place. This time period is introduced here as fault tolerance time. For railway and automotive systems this time is about a few milliseconds. Some of the implemented BIST features must be activated by the user periodically at least once within each fault tolerance time period to guarantee the safety target. To save test-time some tests are performed simultaneously and the CC features become very important because they do not require any interruption of the normal function to detect faults.

Module specific fault detection measures comprising special hardware for CC and BIST have been developed which will be described in the following.

3.1 RAM

There are many testing methods for static RAMs using different fault models [4],[5]. Because of the short fault tolerance time it is not possible to repeat these tests every few milliseconds.

Therefore, CC is a necessary feature for the RAM and for this reason the data and address paths are expanded by a single parity bit. The implemented structure is similar to [6] with some improvements: the well-known self-testing checker of Toy [7] is used to detect word line and decoder faults. A row shift register is not needed for on-line tests.

To detect bridging faults on the logic level the solution proposed in [8] uses a proper sizing of the drivers where a short of its outputs always results in a definite logical high or low level. This is only true for hard shorts. For resistive shorts any voltage level x can appear at each of the shorted nodes. Different gates connected to this node can detect different logical levels (See Fig. 2: inverter and upper AND gate). In this particular case the result can vary depending on the input pin of the AND gate used.

In the design of the RAM decoder, special measures are used for the detection of bridging faults resulting in an arbitrary voltage level. The detection of shorts which could cause such an arbitrary voltage level is almost always ensured by the CC methods implemented. There are only a few nodes for which this is not possible. These nodes are the (inverted or not inverted) address lines which are used for decoding word lines. As shown in [[9], [13] specially sized inverters as level translators are implemented between these points and the address-code checkers to detect the shorts on the logic level - either by the word line checker or by the self-checking address-code checkers [21]. In the example shown in Fig. 2 the word-line checker (as realized in [7]) detects the faulty activation of the additional word line reliably because its addresses differ just in one bit. The word line checker has no chance to detect the fault if the false word line has been activated and the correct one has not. In this case the code checker detects the fault because of the clearly different threshold values of the level translators in comparison to all inputs of the decoder AND gates.

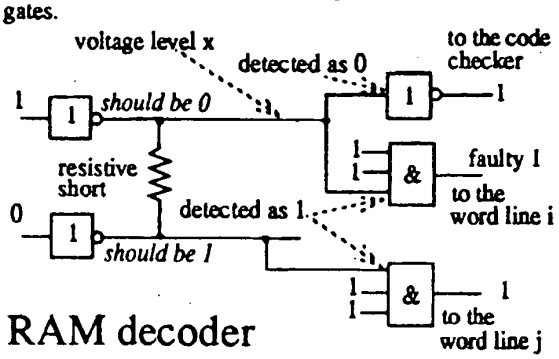

Figure 2: Detection of different logic levels at the same node with voltage level x

Some additional topological layout rules [9], [10] complete the efforts for detection of all single realistic CMOS defects (including common-cause faults [11]). The only necessary condition for a specific full single fault detection is the activation of all word lines to detect latent faults before a second fault appears. If this is not the case, the second fault could mask the detection of the first and vice versa. This activation of all word lines is necessary because of the self-testing-only property of the word line checker. In addition an efficient power-on test is used to detect all multiple faults. Similar to [12] the implemented checkers support the power-on test and shorten the required test time considerably.

3.2 CPU Data Path

A common BIST concept for the data path is shown in [14]. For on-line test purposes the contents of the Register File (containing the program counter and other vital information) has to be saved before each periodical BIST is applied. This could not be performed for the frequently necessary BIST cycles so CC measures similar to those used for the RAM are needed. The classic solution with arithmetic coding for the adder is shown in [15] (Residue Code) and is further developed in [3] for the Bi-Residue Code. Other coding techniques are investigated in [16][17][18]. All these codes require a considerable overhead for an 8-bit data word. It is desirable to reduce the overhead to one additional parity bit as has been done for the RAM. Unfortunately, the parity code is not an arithmetic code. This means that the addition of two code words does not always result in a code word. This problem can be solved by parity prediction as shown in [19]. For the fail-stop controller an approach was chosen following the solution in [20]. In order to preserve the possibility to detect shorts at the logic level our resulting structure is

different - no differential logic realization is used. For CC purposes it must be investigated whether or not a faulty signal level can influence more than one result bit to remain undetected for the code checker. Only one point for each operand bit is very sensitive to shorts when considering common-cause faults - the branching point for the duplicated carry generation. Three specially sized inverters as level translators are inserted at these points as shown in Fig. 3. The sizing is done so as to guarantee the relation between the inverter threshold values under all conditions: vt1 < vt2 < vt3.

In this way the carry *and* the duplicated carry can not detect a false logical value at the operand bit i without detecting the false value by the code checker. In order to detect also line opens at the operand bit i, a layout rule for routing is added. This rule provides the connection first to sum/carry generation, second to the code checker and third to the duplicated carry generation. In the case of a line open at any point of the line in question this fault will be detected with 100% reliability.

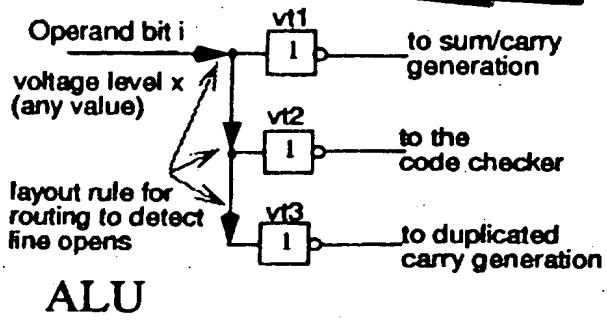

Figure 3: Short detection at the logic level with the inverter threshold values vt1 < vt2 < vt3

Other features realized within the ALU are the checking of the register read and write signals concerning its 1-out-of-n property and in the same way the 1-out-of-m condition of the instruction decoder select lines. In this way a very high fault coverage can be achieved even for non-classic faults.

In [22] a new simulator is presented which can handle shorts at the logic level. For the used fault model at the gate level a bridging 0 or a bridging 1 is assumed if the hard short results in a corresponding logic level. In all other cases a bridging unknown is assumed. We used this simulator to determine the fault coverage for shorts within the ALU.

3.3 CPU Control Path

For the CPU control path the proposal of [23] is implemented to supervise the program flow under user-defined software control. By means of a specially developed assembler, target signatures for parts of the application program can be calculated during assembly and inserted into the normal program flow. Special instructions control the signature monitoring: the signature can be cleared, set, preserved, adjusted and compared.

In addition, all control signals for the data pathware parity coded. The correct codes are checked in the ALU.

3.4 Peripheral Modules

For all peripheral modules the data path is coded by the same parity code as used for the RAM and the ALU. The peripheral modules can be on-line tested simultaneously by means of logical BIST or I_{DDQ}-Test. The test structure is shown in Fig. 4 and the test procedure is applied under user software control.

The implemented test controller can be either activated by the CPU or by the standard IEEE 1149.1 TAP controller as also used in [24]. In both cases the test controller sends a break signal to the CPU and after finishing the break routine the CPU stays in a wait state until the break signal becomes inactive after performing the selected test. For the BIST pseudo-random test generators (LFSRs) in the CPU are used and the bist_ok signals are generated by signature registers (MISR) [25][26] each within the module under test (MUT). The pseudo-random data are transmitted using the existing data bus. The random pattern BIST approach was chosen because of the better fault coverage for non-target faults [27].

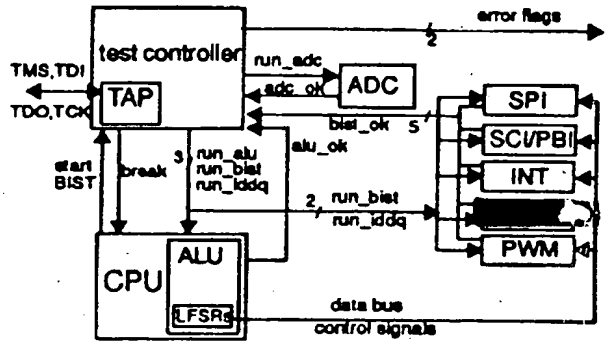

Figure 4: Structure of BIST and I_{DDQ}-Test

For the same reason a part of the same pseudo-random pattern is used for the I_{DDQ}-Test without preselection of efficient test vectors. We did not need the selection method as shown in [28] because of the on-line measurement of the current in 301 very small partitions (See 5.2.) and the high sensitivity of the current sensor [29],[30]. Therefore the test time can be very short - the clock frequency is reduced to only a quarter even for the measurement of high resistive shorts [31]. For this reason we can apply an unselected higher number of test vectors in confidence to a high fault coverage to be reached. To get different sensitivities the duration of the measuring phase can be changed under software control by the user. The importance of I_{DDQ}-Tests does not need to be explained. For CMOS circuits the gate oxide short is one of the most likely defects. The defect model is explained in detail in [32].

3.5 Analog to Digital Converter

For the ADC we use a CC approach for a plausibility check after each conversion. The conversion is done using a successive approximation algorithm for the capacitor array. After switching an additional small test capacitance twice when the conversion procedure has been completed the

comparator output is observed. In this way we can determine if the comparator input is near the threshold value. Thus each conversion result can be checked on-line easily. The BIST of the digital part of the ADC is implemented with the same sequence as for the conversion but with pseudo-random comparator inputs. The correct decision of the comparator to the pseudo-conversion result is checked after that. For the analog part a BIST without need of an external analog value is possible. The result of the BIST must be equal to half the voltage range of the ADC input pins.

3.6 Power Supply Monitor

The power supply is observed by means of a special monitor which determines and stores all deviations of the specified values. The parameters of this circuit using a resistor network, a bandgap reference and operational amplifiers are independent of the technological parameters apart from matching of the resistor values.

3.7 Temperature Monitor

A temperature monitor as shown in [33] has been implemented. The current sensitive oscillator is supplemented by means of frequency dividers and counters to get a digital representation of the oscillator frequency. The influence of the technological parameters to the oscillator frequency should be investigated by means of measurements. Up to now only a few chips from one lot have been investigated. They show an acceptable small standard deviation of about 2.5%. If higher precision is needed a calibration of the temperature monitor can be achieved digitally.

3.8 Test Controller

The implemented test controller is activated by the CPU or by the included Boundary Scan TAP controller. It follows a partial BIST concept which means that either a selected BIST or a full BIST can be performed by the user. Whenever activated by the user software the test controller interrupts the action of the CPU and controls the BIST. Its activity is shown by a special output signal "test" and the functional pins enter an inactive state. The TAP implementation is carried out as shown in [34]. By the use of an extended instruction register several BISTs with different features can be applied. If a fault has been detected either during BIST or I_{DDQ}-Test, the double rail error outputs signal this fault permanently. In the same way the CC circuits show an error using self-testing and self-checking checkers.

4 Test Concept

The implemented test concept provides a hierarchical structure of several test shells with a known fault coverage. First the user runs a set of power-on tests when the system is started, according to recommendations in an application handbook, to check whether the controller, its peripheral modules and the memories are fault-free. Then the normal operation of the controller is interrupted periodically, depending on the system-specific fault tolerance time, to perform on-line logic and/or I_{DDQ}-Tests as well as some functional tests. In this way the occurrence of faults during operation is indicated at the latest by the end of the fault tolerance time of the system. The CC measures, however, flag the occurrence of a fault without interrupting the controller's operation immediately when the fault becomes effective, i.e. when it has an effect on data, instructions or addresses.

This test concept is based on a module-specific, bottom-up approach to cover physical defects. The target fault classes for the on-line tests are beyond the classical stuck-at model. Especially shorts and opens, which are typical defects in CMOS technologies must be considered. That is why pseudo-random patterns are used for on-line tests because they detect more of these non-target faults than deterministic patterns, generated for stuck-at faults [27], [50]. The BIST or I_{DDQ}-Test is executed simultaneously for different peripheral modules as shown above.

On-line logic and I_{DDQ}-Test are invoked by user's software and are controlled by the Test Controller module of the FSC while the operation of the CPU is disabled to run the tests. The occurrence of a fault during the self-tests is flagged in the respective module, captured by the Test Controller and results in a dual rail error signalizing on external pins of the FSC. In this way the system notices a defect in the controller and runs into a user-defined safe state.

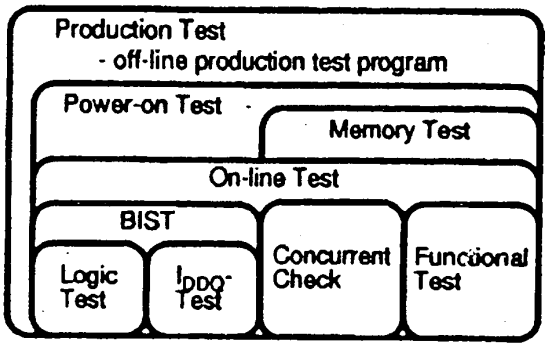

Figure 5: Test shell structure

Additional fault detection is given by the CC measures that are implemented. If an error is detected in this way the same error signaling used in the BIST is used to highlight the faulty state of the controller.

Although the self-test ability is the basic property of the Fail-Stop Controller, additional functional tests under user responsibility are necessary to check circuitries that are not self-testable, for example, peripheral pins of the communication interfaces, the Interrupt Controller and the ADC input channels.

A considerable share of these functional tests is part of a typical application program that uses the designed functionality of the incorporated modules of the controller. Furthermore, several safety measures like the use of a signature analysis of the CPU-opcodes, parity checks for external ROM and RAM accesses, dedicated algorithms for power-on tests of the internal and external memories are recommended in an application handbook. Their use in conjunction with the Concurrent Checking features will detect most of the remaining faults after BIST (Fig. 5).

The safety level of the system is ensured by periodical

self-tests under user control. The specific fault coverage of the different test shells is determined by fault simulation.

5 Aspects of Design Flow Management

In the past, "safety" was a property attributed to systems containing several chip components. An in-depth safety assessment of these components fails because in terms of safety engineering methods we are dealing with black boxes. Nowadays, one chip can represent a very complex system where the safe data processing has to be taken into consideration inside. Hence, safety engineering methods and chip design techniques have to be combined to constitute an extended design environment which is able to support safety requirements in a consistent design flow.

Seeking (self)test synthesis tools capable of managing CC and BIST as a feature on a high level of abstraction we find methods treated in the literature [38][39][40][41], but up to now they are not commercially available. Nevertheless, it was a goal of this project to manage the design of the FSC as an industrial ASIC design. The basis of this design methodology has been already outlined in [42]. The backbone of the design environment is based on commercial tools supplemented by specially tailored tools to meet the requirements constituted by CC, BIST and on-line I_{DDQ}-Test.

Our general approach solving the testing problems follows the principle „divide and conquer": the design is partitioned into testable modules of smaller size which are controlled by a top level test regime supervised by a test controller as shown above. Because the top level BIST architecture has been defined manually this paper presents the design process and tools developed to rapidly include CC, BIST and on-line I_{DDQ}-Test starting from the VHDL behavioral description on register transfer level down to the physical layout.

Since different methods are used to implement fault detection measures, the FSC can be divided into:

1. Modules with very specific hardware test features, designed in a fully customized style and handcrafted layout (e.g. RAM);
2. Modules where we followed a top-down design method taking into consideration all module specific test requirements during the first synthesis scenario what we call "1-pass synthesis";
3. Modules under logic stuck-at BIST and I_{DDQ}-Test. Particularly the I_{DDQ}-Test structures are inserted after the first synthesis pass by specially tailored tools.

The latter two methodologies will be presented in the next sections with a special focus on the I_{DDQ}-Test.

5.1 1-pass synthesis scenario

Based on the functional and test architecture the module designer has the full responsibility for both the pure functional view and the so called test view of the module. In accordance with [40] the introduction of a test view in addition to the more classical views helps to make all the required information available in order to handle the test-related functionality during the design process. In our case the test view is a repository containing the information needed for the CC and BIST, for example: parity bit, special cells supporting testability, timing information related to the BIST, module specific signature. We must notice that modules which only support test functionality, for example the test controller, have a nearly empty functional view. The sum of both views specify the module in its entirety and constitute the basis to describe all functional blocks of the controller including the testbenches in the hardware description language VHDL. To achieve gate level we use a commercial synthesis tool. Every module designer can verify the actual gate level representation in the mixed-level simulation environment. To speed up the simulation only the current module is represented on gate-level whereas the other modules as well as the testbench are described on the system level. According to a modern design flow a very close link to the standard cell layout tool is established comprising various features such as timing driven layout, clock-tree synthesis and the extraction of routeability costs from the layout.

Up to this point, with the exception of the test view, this is an ordinary design flow. The question is could the test structures required by CC and logic stuck-at BIST be inserted within the first synthesis scenario.

For on-line fault detection purposes redundancy is needed. Depending on the kind of test, this useful redundancy can be achieved by coding or duplication redundancy and circuitries dedicated to control the BIST. All these additional circuitries can be faulty. To ensure the detection of faults in these parts, self-testing and self-checking circuits are preferred. Using a VHDL synthesis tool the implementation of coded data processing at the behavioral level is possible. In the same way the control signals for the BIST and I_{DDQ}-Test can be synthesized easily.

Special hardware components are implemented for measurement and signalling purposes. These are voltage level sensitive circuits to detect bridging faults and self-checking checkers. To enforce the correct implementation by synthesis a more structurally oriented hardware description, carefully constrained, is necessary. These components must be labelled "don't touch" to prevent an optimization by the synthesis tool at the expense of the safety features. Ignoring these design rules means that the self-testing or self-checking behavior can be affected, the logical detection of bridging faults can not be ensured or the layout rules for fault avoidance may be corrupted. A more detailed description of these points can be found in [37].

The term "1-pass synthesis scenario" introduced by [38][41] is defined as a "synthesis of an optimized logically-correct testable module in one iteration, starting from an HDL description". Using commercial synthesis tools test requirements can not be described on a higher level of abstraction. A more structural description using specific test auxiliary cells is required which looks more like a workaround. But nowadays this is the only way to combine a modern top-down design flow with test structure synthesis in a 1-pass synthesis scenario. Many of the rules related to CC and BIST can only be described on a lower level of abstraction, in many cases even on gate level, of the design. Coding becomes excessive and prone to errors. Therefore, progress is needed to express test requirement

on a higher level of abstraction.

5.2 I_{DDQ}-Test Structure Insertion - 2-pass Test Synthesis

Despite the 1-pass test synthesis the necessary circuitries for an on-line I_{DDQ}-Test have to be inserted by an additional design step. I_{DDQ} is not a subject we are dealing with when we elaborate the logic behavior of a digital design. It is beyond the scope of circuit synthesis and validation no matter on which level. No support is offered by commercial tools and a full integration of appropriate tools can not be expected in the near future. The only way to integrate the I_{DDQ}-Test circuitries automatically is to modify an optimized logically-correct netlist of a module and make sure that all non-avoidable side effects do not seriously affect the specified performance.

The advantages of the I_{DDQ}-Test in terms of defect coverage are widely accepted (e.g. [43][44][45]). The special dynamic Built-In Current Monitor (BICM) used is particularly sensitive to resistive shorts and can detect such defects (like high-resistance gate oxide shorts) which have no present impact on the logical function. By this means a very effective failure prevention is possible. Therefore, the I_{DDQ}-Test technique is an important part of the overall test strategy of the FSC. On the other hand, the incorporation of I_{DDQ}-Test circuitries has the most significant impact on the design flow. In the next section we want to present a method for I_{DDQ}-Test insertion which is fully compatible with the CMOS process used in a standard-cell-based design for industrial use.

The current sensor shown in Fig. 6 has already been presented in [31][46]. For details see also [32]. According to the general test concept the current sensor must be tested as well. Applying a logic high level at the T_{test}, a leakage is simulated and the functionality of the current detection technique is tested.

The discharge time of the virtual-VDD node depends basically on the parasitic capacitance of the MUT including the complete well-substrate capacitance. This overall parasitic capacitance of a module in a modern highly integrated circuit is too large and the test time requirements can not be met. Therefore, the MUT has to be partitioned into sub-modules which are linked by a common evaluation circuit. The number of partitions depends on the size of the partition, it means size in terms of the total parasitic capacitance. For all partitions we have to make sure that the term $C_{parasitic}*R_{leakage}$, ($R_{leakage}$ - the resistance we want to detect) is less than the test cycle time T_{test}. This is the first and the most important constraint in order to meet the desired test time requirements.

A very important problem we have to deal with is the performance loss caused by the bypass transistor which can be regarded as a serial resistor in the power supply line. This affects the speed of the switching gates of the MUT. Since we have to distinguish between the working phase and the test phase, the bypass transistor size has no impact on the test resolution. Thus, performance loss and the size of the bypass transistor can be traded off separately from all other design considerations. This is the most important advantage of the I_{DDQ}-Test technique presented here. Nevertheless, the performance loss has to be kept small and has to be balanced by an appropriate partitioning. Our approach to partition the circuit is that we use only one current sensor cell with a fixed size of the bypass transistor and we vary the count of partitions and the permutations of cells. The constraints of the partitioning cost function can be summarized as follows:

1. Make sure the condition:
   $C^k_{parasitic}*R_{leak,max} < T_{test}$ for each partition k!
2. Try to balance the transient current for all partitions taking into account all possible functional states!

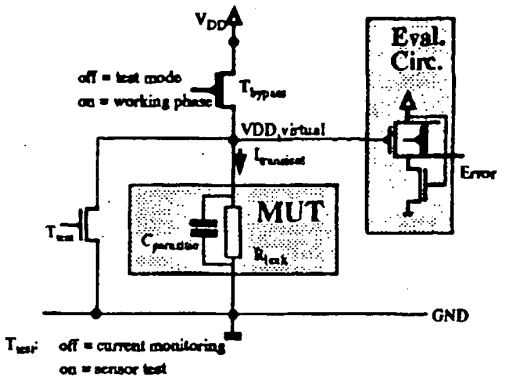

Figure 6: Dynamic current sensor

At the present time no solution exists which will allow us to fulfill the second requirement. Hence, the more practical solution is that we minimize the difference between the maximum current request of the partition $I_{transient,peak}$ and the current which can be granted by the bypass transistor. To get $I_{transient,peak}$ we transform the whole pull-up circuit of the partition into one effective PMOS transistor $T_{effective}$ taking into account all parallel and serial transistor structures. The width of $T_{effective}$ is a measure for $I_{transient,peak}$ and gives an upper bound because we base our calculations on the hypothetical assumption that all PMOS transistors switch simultaneously.

The relation between the width of the bypass transistor and the width of the effective PMOS transistor is the only degree of freedom given in the following equation [35].

$$\frac{Width(Tr_{bypass})}{Width(Tr_{effective})} = K_W$$

As partially discussed in [47][48], we developed a placement-oriented partitioning which takes place at the back-end of the standard cell design flow. A requirement for practical use was to adapt the BICM-cells to a standard cell design and to make all additional design steps compatible with our design environment.

Fig. 7 shows an outline of the placement and power routing in a standard cell design block including the I_{DDQ} sensor cells. The global power routing is vertically oriented. The gates of a partition are located on the right and left side of the sensor cells linked by the horizontal virtual power supply lines. The partitions are separated by spacer cells.

The partitioning methodology briefly described in the previous section was implemented in a sensor-placement

tool which fit well into the commercial Place & Route Tool.

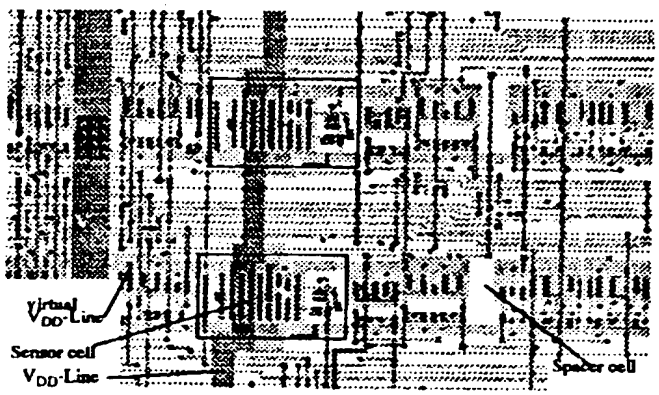

Figure 7: Standard cell design including I_{DDQ}-Test

As depicted in Fig. 8 this tool establishes an additional design step after the preplacement of all logic cells according to the synthesized netlist which comprises the following features [49]:

- Parsing netlist and library information
- Parsing preplacement information
- Partitioning of the MUT according to the partitioning parameter K_w
- Generation of the evaluation circuit
- Generation of the netlist add-on (sensor cells, evaluation circuit, appropriate interconnects)
- Generation of the final placement (cell insertion of I_{DDQ}-sensors and evaluation circuit)

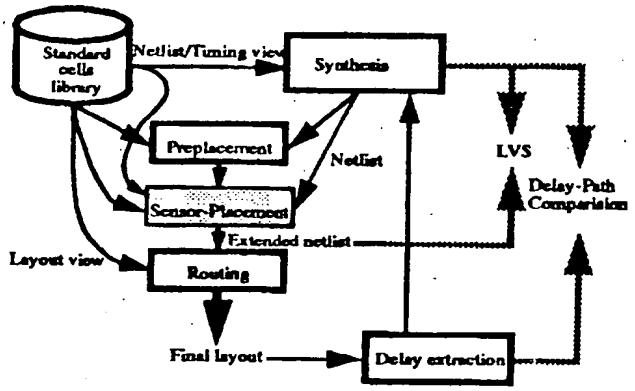

Figure 8: I_{DDQ} place & route flow

Because the partitioning takes place at the back end of the design flow and it can not be performed concurrently with the logic synthesis pass, the netlist is modified by additional instances and interconnects. Therefore, to preserve the flow consistency, the final verification between the physical layout design and the schematic representation of the circuit, usually done by an layout versus schematic check (LVS), has been supplemented by a delay path comparison, see Fig. 8. Within our environment the interconnect delay backannotation is a standard f ature. All possible delay paths are extracted with the associated net names from the final layout. Nets with no representation in the synthesis database refer to the I_{DDQ}-Test circuitries. The detailed elaboration of this information which results in a clearly arranged list of the netlist add-on, we call Delay-Path-Comparison.

As already mentioned above the performance loss is a negative side effect of the I_{DDQ}-Test proposed. Although minimizing the performance loss is a prime objective of the partitioning, the performance validation is an indispensable requirement. The switch-on Drain-Source resistance is unavoidable. Because the circuit performance loss does not depend on the number of cells linked by the current sensor but on the amount of current and thus the number of gates switching simultaneously, the performance degradation problem becomes a dynamic one. The degradation depends on the circuit structure and the input pattern and can not be predicted completely. Because it is impossible to characterize cells in terms of timing, taking into account the dynamic voltage drop effects on current requirements, our Top-Down design methodology is violated. The Top-Down design methodology assumes precharacterized cells with a fixed pin to pin delay.

Therefore, we introduced an accurate dynamic simulation of all effects combined. To deal with complex modules as many as 10000 Gates we needed a high speed detailed electrical simulator. Thanks to the development of such tools we were able to simulate our most critical modules within a reasonable runtime. Assuming a K_w=1 a delay increase of the longest path delay of about 15% has been shown by simulation. This performance loss has to be considered in the synthesis constraining in advance and underlines the necessity of this additional design validation step.

6 Fault Simulation and Safety Estimation

A fault simulation determines the coverage of a test pattern with respect to the chosen fault model for the investigated circuit, expressed in terms of the fault coverage (FC).

$$FC = \frac{\text{detected faults}}{\text{detectable faults}} \times 100\,\%$$

detectable faults = all faults - undetectable faults

The classical approach is to use the stuck-at fault model. For our application this method is not sufficient because the on-line tests of the FSC also have to detect physical defects that are not represented by the stuck-at model to ensure the safety of the controller.

The fault simulator used offers, in addition to stuck-at faults, a treatment of bridging faults (shorts). The latter are considered to be detected by a test vector, provided an I_{DDQ} sensor observes the module under test and the short is activated. Activation means, the considered test vector drives the two affected nodes to different logic levels and hence causes a significant increase of the static current consumption.

A fault simulation was performed for the CPU and such peripheral modules that realize the functionality of the

controller. Other modules, that are used for test purposes only, like the Test Controller, the Signature Analysis module or the Boundary Scan cells have not been investigated. The design and the test concept guarantee that faults in these modules either will be detected by BIST, Concurrent Checking or signature errors or they do not have any effect on the operation of the controller. The fault coverage of the memories is determined by means of fault grading, the ADC is checked by a separate BIST for the analog and digital part and by a plausibility check [13].

First, a fault simulation of stuck-at faults was performed for every single set of test patterns. Table 1 shows the results in detail [36] [51] [52]. The fault coverage for the implemented logic BISTs of the peripheral modules using pseudo-random patterns is in the range of 53% to 69% whereas the ALU is covered almost 100% by the logic test, also because of its built-in CC measures.

In a second step the remaining stuck-at faults after the logic BIST are transferred into shorts by regarding them as shorts between the affected node and VDD or GND, respectively. Then a fault simulation of the I_{DDQ}-Test is performed and the fault coverage of the peripheral modules increases by a value of between 22% and 37% when the I_{DDQ}-Test is executed in addition to the logic test. In this way a total coverage of approximately 90% regarding stuck-at faults is achieved by the BIST of the peripheral modules.

A closer look at the remaining faults after BIST yields a classification of several types [51]:

a) Not detected faults are located in circuits, that are not activated during BIST, for instance the address decoders of the peripheral modules. Simple functional tests like CPU read or write accesses to the special function registers in the modules lead to the detection of these faults by Concurrent Checking. The detection of this kind of fault is given automatically at each access.

b) In the first silicon only one out of several dozen I_{DDQ} sensor cells per module is self-tested, resulting in a considerable amount of remaining faults. In the second silicon all sensor cells are self-tested. Thus, these faults will be detected [31].

c) The treatment of faults that are detected by Concurrent Checking exposed the problem that there is no software tool like a fault simulator available to prove which faults are detected by these measures. To cope with this situation a careful fault grading was performed, especially for the RAM 4k and the bus-interfaces of the peripheral modules. Such faults that are definitely detected by Concurrent Checking or are avoided by special layout rules are removed from the list of remaining faults.

d) Other types of faults can also be excluded, because they affect parts of circuits, which are accessible in the production test mode only and do not disturb the operation of the controller - furthermore, they can not affect the normal operation without observation. Thus, they are not relevant to the user.

A common value for the increase in fault coverage - achieved by Concurrent Checking, functional testing and fault avoidance - is given in the 4th column of Table 1 for each module, although it is possible only for functional testing to determine this value by fault simulation [36]. The results after fault grading show a coverage of more than 98% obtained for the peripheral modules and the CPU-ROMs in the second silicon, while the ALU with its registers is covered 100%. For the latter not only the BIST has a very high FC. Also the CC measures implemented lead to nearly the same fault coverage as the BIST. The RAMs are also fully tested in on-line mode.

Module	Production Test			
	On-line Test			Test Mode
	BIST	I_{DDQ}	CC	
INT	69.1	91.3	98.5	98.9
PWM	53.4	90.2	98.4	98.4
SCI	66.2	89.3	98.7	98.8
SPI	58.3	85.6	98.6	99.0
PBI	64.2	92.3	98.1	98.4
WD	57.4	87.7	99.5	99.7
ALU+REG.	99.7	-	100	-
CPU-CTRL	-	-	80.4	-
CPU-ROM	-	-	98.9	-
CPU-RAM	-	-	100	-
EBI	-	-	73.4	-
RAM 4k	-	-	100	100
Total excl. RAM 4k			98.19	98.28
Total			99.72	99.74

Table 1: Fault Coverage in %, stuck-at faults

The value for the control circuit of the CPU is relatively low because a fault grading is not performed for this module and the fault coverage depends on the application program used if the supervising of the program flow by the signature module is included. Furthermore, a considerable part of the External Bus Interface (EBI) is designed for production test mode accesses only. Since the functional patterns do not activate this circuitry and a set of test mode patterns is not simulated, the fault coverage of the EBI is

also relatively low. It is also estimated to reach at least 95% by additional patterns and fault grading for both modules. Nevertheless, further calculations do not take estimated values into account.

The total fault coverage is determined on the basis of the module specific fault coverages[36]. Because of the 4kByte-size of the RAM and its immense influence on the fault coverage two values are shown - with and without RAM. The value of 98.19% for the on-line tests of the CPU and the peripheral modules is calculated separately without RAM. The value increases to 99.72% when the RAM is included.

Only on the production tester can additional test mode pattern be applied. At this time an entire RAM March test according to [53] as well as all power-on and on-line test routines are executed to ensure a proper quality level of the controller. In comparison to the on-line tests there is only a slight increase of 0.02% in fault coverage visible for the production test. Thus, the on-line tests achieve almost the same test quality as the production test.

6.1 Inductive Fault Analysis

In order to cope with physical defects in CMOS technologies that are not represented by the stuck-at fault model, a specialized tool was developed to perform an Inductive Fault Analysis. This tool works like a design rule checker. It uses layout and technology information to establish a list of possible shorts in the considered design, caused by spot defects between adjacent layers or structures, respectively [54] [55]. This procedure allows the exclusion of shorts between two nodes of a netlist which have no adjacent physical layers in the layout of this circuit. Although such a list is usually quite long, the number of shorts under consideration is reduced dramatically in comparison to a list with shorts possible between all nodes of a netlist, which can not be processed.

For several modules an extraction of possible shorts is performed, followed by a fault simulation [51]. The results are depicted in Table 2.

Module	#faults	FC	Comments
INT	33 252	87.2	I_{DDQ}-Test
PWM	29 556	85.6	
SCI	53 831	83.8	
SPI	25 727	83.5	
PBI	73 170	86.7	
WD	17 903	71.9	
ALU + REG	63 025	98.6	Logic BIST

Table 2: FC of shorts in %, without fault grading

The coverage of the I_{DDQ}-Testing for shorts is in the same range as the coverage of the logic BIST, followed by an I_{DDQ}-Test for stuck-at faults. Because the investigation of the remaining faults is not as easy as for stuck-at faults, the detection of shorts by Concurrent Checking and functional testing is not evaluated. It is expected, that the coverage of shorts is increased by these measures in the same way as for stuck-at faults.

Since the fault simulator is only able to evaluate a pattern if an I_{DDQ} sensor observes the module under test, a set of tools called CAT-TOOLS was developed by the University of Hannover [22] [52] [56]. It supplies a fault simulator that deals with shorts on the gate level. Although the effect of a short between two nodes is an unknown logic level, the simulation is based on the assumption, that all the gates, driven by these nodes, will evaluate the resulting level uniformly either logical low or high. Then the short is regarded as detected by a pattern, if the pattern detects both the faulty low and high level on these nodes.

The CPU data path is analyzed with the CAT-TOOLS and the result is a coverage of 98.6% for shorts [36]. This corresponds with the value for stuck-at faults and proves the detection of technology-relevant physical defects by the built-in test measures.

6.2 Risk Rate Calculation

A risk rate calculation is based on the assumptions, that a component is fault-free after production test and single, independent defects will occur during operation with a constant failure rate λ and with the same likelihood, whereas the likelihood of the occurrence of two independent single faults within a certain time is negligible [52] [57]. On the other hand all defects are considered to be critical in a safety-critical application, hence all defects must be detected by proper measures within the fault tolerance time of the system. The risk rate is an expression of the likelihood, that such a fault occurs, that is not detected within the fault tolerance time. The risk rate λ_r of the FSC is defined by:

$$\lambda_r = (1 - FC_{on\text{-}line}) \cdot \lambda$$

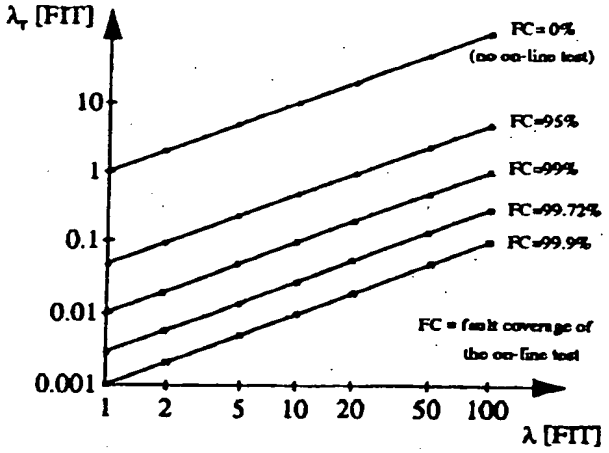

Figure 9: Risk rate vs. failure rate (logarithm. scale)

While the failure rate λ of an Integrated Circuit depends on the fabrication process and is usually not exactly known in advance, the fault coverage of the on-line tests $FC_{on\text{-}line}$ is

the key parameter in the design process to guarantee the necessary safety level.

Based on the results of the fault simulation for stuck-at faults the risk rate of the FSC can be determined easily. Provided the failure rate λ is between 1 FIT and 100 FIT (1 FIT = $10^{-9}h^{-1}$) the risk rate will be in the range of 0.0028 FIT and 0.28 FIT when the fault coverage $FC_{on\text{-}line}$ is 99.72%. Fig. 9 shows the dependence of the risk rate on both the fault coverage and the failure rate

7 Results

Silicon of the final version of the Fail-Stop Controller AE11 containing about 400,000 transistors is available. The full functionality of the chip has been verified. The area overhead needed for the BIST, I_{DDQ} and CC-features is about 35%. The performance loss caused by the delay of the I_{DDQ} bypass transistor and the parity generation circuitry is less then 10% even if some signal paths are delayed by more then 15% as shown above and in [31],[35]. This is due to the influence by the I_{DDQ} and CC measures implemented only in one part of the critical path. The stuck-at fault coverage for the BIST and I_{DDQ}-Test reaches 99.72% after fault grading for the on-line test as shown in [36] in detail.

A very high fault coverage could be shown also for the CC circuits: 99.7% for the CPU data path and more than 99.8% for the RAM. By adding a functional pattern these fault coverages can become 100%. This FC is not restricted to stuck-at faults because of the error detection property of CC and the implementation of special circuitry for logical detection of resistive shorts.

8 Conclusions

A great variety of different measures was necessary to ensure a high on-line fault coverage, acceptable area overhead and low performance loss. The implemented additional circuitry needed for I_{DDQ}, BIST and CC is different for the modules of the controller. To preserve the stored data within RAM and CPU registers the non-destructive CC is preferred for the on-line test of these modules. Even to detect bridging faults at the logic level during normal operation, some special level translators have been implemented in these modules. Special layout rules avoid common-cause multiple faults.

For the peripheral modules logical BIST and on-line I_{DDQ}-Tests have been implemented. The on-line I_{DDQ}-Test at 4 Mhz for resistive shorts up to some dozens of kΩ as proposed in [29] and its realization in the Fail-Stop Controller shown in [31] was a challenge.

To supervise the program flow a special assembler has been developed to allow a pre-calculation of a flow dependent signature. This signature can be compared at runtime by means of special hardware (signature analysis module) under software control. The very high on-line fault coverage indicated was brought about by the implementation of recommendations from the application handbook for the user. If some of the suggested tests are performed at least once within the fault tolerance time of the system, the applied measures for on-line fault detection will guarantee the necessary safety level of the single-channel solution of a controller for safety-critical applications, proved by a risk rate calculation. In addition to the implemented CC either the I_{DDQ}-Test or the BIST of the peripheral modules has to be performed alternatively within the fault tolerance time. In addition, a full BIST must be performed at hourly or daily intervals to detect latent faults.

To signal faults two independent special error pins are used which must always have different logic values changing at least once within the fault tolerance time. These error pins show the summarized results of all fault detection measures using self-checking checkers.

The implementation of CC, BIST and I_{DDQ}-Tests was an innovation when compared with classic design aims directed to area reduction and performance enhancement. Our approach was to find a way to describe (self)test requirements on a higher level of abstraction. The lack of suitable tools was evident during the entire design phase in this project. Nevertheless, CC- and BIST-related circuitries have been implemented in a so-called 1-pass design method. But special CC- and BIST-features had to be described on a lower level of abstraction at the expense of design efficiency.

The on-chip I_{DDQ}-Test methodology developed in this project is neither fully CMOS-compatible nor does it fit well into a standard cell design style. It has been implemented by introducing an additional design step after the 1-pass synthesis scenario. But this design step has a tremendous impact on an ordinary Top-Down design flow. A tool which interfaces all major design steps has been developed in order to insert the I_{DDQ}-Test cells and the associated interconnects. Using this tool we are able to maintain the design efficiency comparable to an ordinary standard cell design flow.

9 References

[1] Osseiran,A. Nicolaidis,M. et al.: Design of a Self-Checking Microproc. for Real-Time Appl., IFAC SAFECOMP Como, Italy, 10/85

[2] Chaumotet,G. Nicolaidis,M. et al.: MAPS: A Safety Microcontr. Dedic. to the Railway Control, IMAG/TIM3 Univ. of Grenoble, 90

[3] Holzapfel,H.P. Horninger,K.H.: Fault Tolerant VLSI Processor, 3rd international GI/ITG/GMA-Conference, Bremerhaven 9/89

[4] Goor, A.J. van de: Testing Semiconductor Memories, Theory and Practice, Wiley & Sons 1991

[5] Dekker,R. Beenker,F. Thijssen,L.: A Realistic Fault Model and Test Algorithms for Static RAMs, IEEE CAD 6/90, 567-572

[6] Savin,H.V. et al.: Design for concurr. Error detection and Testability in Storage/Logic Arrays, IEE JSSC vol.29 no.7, 1994, 770-779

[7] Toy, W.N.: Modular LSI Control Logic Design with Error Detection, IEEE Trans. on Computers, C-20, 2/1971

[8] Nicolaidis, M.: Shorts in self-Checking Circ., ITC 1987

[9] Böhl, E.: Self-checking RAM- modules under Consid. of realist. faults, 7th Worksh. ITG/GI FG Hannover,2/95

[10] Nicolaidis,M. Courtois,B.: Layout Rules for the Design of Self-Checking Circuits, Proc. IFIP Tokyo 1985

[11] Smith,A.M. Watson,I.A.: Common Cause Failures - A Dilemma In Perspektive, Proc. ann. Reliablity and Main-

tainab. S. 1980, 332-339
[12] Gupta,S.K. Pradhan,D.K.: Can Concurrent Checkers Help BIST?, ITC 92, 140-150
[13] Böhl, E. et al.: The Architecture of the Fail-Stop Controller AE11, 3rd IEEE Int.. On-Line Testing Workshop, Crete, Greece, July 1997
[14] Gizopoulos, D. et al.: An Effective BIST Scheme for Datapaths, ITC 1996, pp 76 - 85
[15] Peterson,W.W.: On Checking an Adder, IBM Journal April 1958
[16] Lo,J.Ch. Sun,S.Y. Daly,J.C.: A Concurrent Error Detection IC in 2µm static CMOS Logic, IEEE JSSC, May 1994, 580-584
[17] Iha,N. Wang,S.J.: Design and Synthesis of Self-checking VLSI Circ, IEEE Trans. on CAD, June 1993 878- 887
[18] Fujiwara,E. Pradhan,D.K.: Error-Control Coding in Computers, Computer July 1990, pp. 63-72
[19] Fujiwara,E. Haruta,K.: Fault-Tolerant ALU using Parity-Based Codes,Trans.o.t. IECE o. Japan, E64, No.10, Oct.1981
[20] Nicolaidis,M.: Efficient Implement. of Self-Checking Adders and ALUs, FTCS-23, Toulouse 1993, 586-595
[21] Wakerly,J.F.: Error Detecting Codes,Self-Checking Circuits & Applicat., Elsevier North-Holl. 1978, 54-223
[22] Mahlstedt, U., Alt,I J.: Simulation of Non-Classical Faults on the Gate Level - The Fault Simulator COMSIM, ITC 1993, pp. 883-892
[23] Wilken,K. Shen,J.P.: Continous Signature Monitoring: Low-Cost Concurr. Detect. of Proc. Control Errors, IEEE CAD, 6/90,629-641
[24] Reed,D. et al.: Improving Board and System Test: A Proposal to Integrate Boundary Scan and I_{DDQ}, ITC 1995, pp. 577-585
[25] Savir,J. Bardell,P.H.: Built-In Self-Test: Milestones and Challenges,VLSI Design 1993, Vol.1 No 1, 23-24
[26] Bardell,P.H. et al.: Built-In Test for VLSI: Pseudorandom Techniques, J.Wiley & Sons, 1987
[27] Favalli,M. Olivio,P. Damiani,M. Ricco,B.: Fault Simulation of Unconventional Faults in CMOS Circuits, IEEE CAD 5/91, 677-682
[28] Mao,W. et al.: QUITEST: A Quiescent Current Test. Methodology for Detect. Leakeage Faults, Proc. Int. Conf. CAD, 280-283,1990
[29] Kesel,F.: Selbsttest von nicht-regulären CMOS-Schaltungen in sicherheitskritischen Anwendungen, Diss., Univers.Hannover, 1994
[30]Kesel,F.: Built-In Self-Test of CMOS Random Logic Using I_{DDQ}-Testing, BIST/DFT Workshop, April, 19.-21.1994, Radisson Resort, Vail, CO, USA
[31] Meerwein,M. et al.: A dynamic I_{DDQ} Monitor Cell with Self-Test for the Fail-Stop Controller AE11, 3rd IEEE Int. On-Line Testing Workshop, 7/1997, Crete, Greece
[32] Segura, J. et al.: A Detailed Analysis of GOS Defects in MOS Transistors, ITC 95, 544-551
[33] Arabi,K. Kaminska,B: Built-In Temperature and Current Sensors for On-Line Oscillation-Testing, 2nd IEEE Intern. On-Line Testing Workshop, Biarritz,July 1996
[34] Pichon,F.: Implementing BIST and Boundary-Scan in a Digital Signal Proc. ASIC for Radiocommunic. Appl., ETC 93, 361-369.
[35] Lindenkreuz,Th. t al..: Design Flow Asp. of the Fail-Stop Contr. AE11, 3rd IEEE Int. On-Line Test. Worksh., 7/1997, Crete, Greece
[36] Seydel,G. et al.: Test Concept of the Fail-Stop Controller AE11 using BIST and I_{DDQ}, 3rd IEEE On-Line Testing Workshop, 7/1997, Crete, Greece
[37] Böhl, E.: Experiences in the Design of a CPU with On-Line Self-Test, 1st IEEE Int. On-Line Testing Worksh.,Nizza July 4-6 1995
[38] Bennetts, B.: Test Synthesis: Towards Higher Levels of Abstract., II Archimedes Workshop: Synthesis of testable circ., Barcelona 2/95
[39] Roy, R.K.: Advantages of High Test Synthesis over Design for Test, Proc. Intern. Test Conf., p. 293, 1995
[40] Landrault, Ch., Flottes, M., Rouzeyre, B.: Is high Level Test Synthesis just Design for Test?, ITC 1995, p. 294
[41] Maxwell, P.C.: The many Faces of Test Synthesis, Proc. Intern. Test Conf., p. 295, 1995
[42] Papanuskas, J., Mößner, B. et al..:A uniform design methodology for application specific digital ICs in automotive applications, European DAC, Brighten, 1995
[43] Perry, R.: I_{DDQ} Testing in CMOS ASICs - Putting in all together, ITC, pp. 151-157, Sept. 1992
[44] Gayle, R.: The Cost of Quality: Reducing ASIC Defects with I_{DDQ} At-Speed Testing, and Increased Fault Coverage, ITC, pp. 285-292, Oct. 1993
[45] Wiscombe, P.C.: A comparison of Stuck-At FC and I_{DDQ} Testing on Defect Levels, ITC, 293-299, Oct. 1993
[46] Kesel, F., Built-In Self-Test mit hoher Defekterfassung durch On-Chip I_{DDQ}-Monitoring, 5th ITG/ GI/GME-Workshop, Holzau, 1993
[47] Lindenkreuz,Th., Ringe,M.: Realisierung eines I_{DDQ}-Tests in standardzellbasierten Designs, 8th ITG/ GI/ GME-Workshop, Otzenhausen, 1996
[48] Lindenkreuz,Th., Ringe,M., Benkoski,J.: On-chip I_{DDQ}-Test as a BIST-technique in a standard cell based design - A review of specific design problems, PATMOS'96 Workshop, Bologna, 9/96
[49] Ringe, M.: Erweiterung einer Standardzell-Designumgebung um Entwurfsschritte zur Integration von Stromsensoren für einen I_{DDQ}-Test, Diplomarbeit, Hannover, 1995
[50] Singh, A. Rasheed, H. Weber, W.: I_{DDQ} Testing of CMOS Opens: An Experimental Study, ITC 1995
[51] Hofmann, D.: Untersuchungen zur FC einzelner Module im FSC, Diplomarbeit FH Reutlingen, 1996
[52] Proc. JESSI AE11 Final Review Meeting, Reutlingen Sept. 3rd, 1996, internal report of Robert Bosch GmbH
[53] Dekker, R. Beenker, F. Thijssen, L.: Fault Modeling and Test Algorithm Devel. for Static Random Access Memories, ITC 1988
[54] Maly, W.: Realistic Fault Modeling for VLSI Testing, 24th ACM/IEEE Design Automation Conference 1987
[55] Jacomet, M. et al.: Layout-Dependent Fault Analysis and Test Synthesis for CMOS Circuits, IEEE Trans. on CAD, vol. 12 6/1993
[56] Alt, J. Mahlstedt, U. Simulation of non-classical Faults on the Gate-Level -Fault Modeling , 11th VLSI Test Symposium, 4/1993
[57] Dilger, E. Manstetten, D.:Computation on the Safety of Identical Dual Channel Controllers, , 5th Workshop ITG/ GI FG Holzhau, 3/1993